Big Data and Computing Visions

www.bidacv.com

Big. Data. Comp. Vis. Vol. 5, No. 1 (2025) 74–93.

Paper Type: Original Article

A Novel Mathematical Model with an LCA-Based Solution Method to Minimize Earliness-tardiness Costs on a Single Machine by Considering Batch Delivery

Zahra Pourali[1,*], Bohlool Ebrahimi[1]

[1] Department of Industrial Engineering & Management Systems, Amirkabir University of Technology, Tehran, Iran; z.pourali@aut.ac.ir; b.ebrahimi@aut.ac.ir.

Citation:

Received: 07 September 2024 *Revised: 11 November 2024* *Accepted: 08 January 2025*	Pourali, Z., & Ebrahimi, B. (2025). A novel mathematical model with an LCA-based solution method to minimize earliness-tardiness costs on a single machine by considering batch delivery. *Big data and computing visions*, *5*(1), 74-93.

Abstract

This paper addresses a practical but complicated version of Just in Time (JIT) problem in which a set of available jobs with known processing times and due dates are processed on a single machine and delivered in batches of arbitrary size. A new mathematical model is developed to minimize the non-convex sum of earliness-tardiness and delivery costs criteria. Due to the limitations imposed by the large size complexity of the proposed model and its nonlinear nature, we use the recently proposed league championship algorithm (LCA) to solve arbitrary test problem instances of the problem on hand. Since LCA works in continuous space, we use several representational schemes to map the solutions generated by LCA to discrete space and compare the output of the algorithm under each mapping scenario. To measure how effective LCA is in comparison with the mathematical modeling approach and other heuristic methods, we use the Lingo system and a discrete version of the Imperialistic Competitive Algorithm (ICA) as the comparator algorithms, respectively. Experimental results show that LCA is strongly efficient and dominates the comparator algorithms. At the same time, the time saved by LCA to report the final output is significant, which recommends the use of this algorithm for other practical optimization problems.

Keywords: Scheduling, Earliness tardiness penalties, Batch delivery, League championship algorithm, Mathematical programming.

1 | Introduction

The competitive environment in production companies makes manufacturers apply proven strategies such as supply chain management, revenue management, total quality management, just-in-time philosophy, and so on to ensure customer satisfaction by producing goods with high quality and also in a reasonable time [1].

Corresponding Author: z.pourali@aut.ac.ir

https://doi.org/10.22105/bdcv.2025.498642.1238

However, the most well-known strategy which exactly emphasizes getting the right products at the right time with the right quantity and quality and with zero penalty costs is the Just in Time (JIT) philosophy.

The basic JIT systems mainly focused on earliness, tardiness, and due date penalties. However, in many cases, we should pay a lot of money to deliver our finished goods to customers. As a result, it is important to think about models that can make a balance between early-tardy penalties and delivery costs. In the modern JIT approach, to be more efficient and applicable to real-world industries, the basic JIT systems considered transportation and delivery costs as new concepts in their modern JIT approach.

On the other hand, with respect to different manufacturing systems in each production environment, different scheduling and production planning for single, parallel, flow shop, and so on environments could be implemented. Nevertheless, in many complicated manufacturing systems, there would be an ability for complex operations of the system to decompose to simpler forms, and each part would be considered a single-machine operation system. Reversely, the knowledge obtained by investigating single machines could be generalized to multiple machines with more complicated environments [1]. In conclusion, in this paper, we concentrate on one-machine scheduling problems, which can provide us with a useful pattern to investigate on multiple machines.

Indeed, the current paper addresses a new version of earliness-tardiness scheduling problems with batch delivery consideration and distinct due dates for the first time, which covers all important concepts that were explained earlier. In other words, we develop a new nonlinear mathematical model to minimize earliness-tardiness costs on a single machine with batch delivery consideration. Due to the limitations imposed by the large complexity of the developed model and its nonlinear nature, we developed a new meta-heuristic solution-based method using the recently proposed league championship algorithm (LCA).

Since LCA works in continuous space [2], we use several representational schemes to map the solutions generated by LCA to discrete space and compare the output of the algorithm under each mapping scenario. In addition, to measure the efficiency of our LCA, the proposed algorithm will be compared with the mathematical modeling approach and other heuristic methods. Therefore, the paper makes several contributions, which can be summarized as follows:

I. Studying a new version of the JIT problem in which a set of available jobs with known processing times and due dates are processed on a single machine and delivered in batches of arbitrary size for the first time.

II. Developing a novel mixed integer nonlinear model for our non-convex objective function with batch delivery criteria.

III. Developing a meta-heuristic-based solution approach by using the LCA algorithm to estimate the optimal solution of the developed nonlinear model.

IV. Extending the LCA algorithm to estimate the optimal solution of discrete problems.

The remainder of the paper is organized as follows. In Section 2, we will review the literature related to our work. Section 3 describes the basic definition of our combinatorial problem and develops a new mathematical model. Our proposed LCA is explained in Section 4. Experimental design and computational efforts are presented in Section 5, and the conclusions are given in Section 6.

2 | Literature Review

This paper addresses a new version of earliness-tardiness scheduling problems with batch delivery consideration and distinct due dates. A set of n jobs $J_1, J_2, \ldots, J_n$ with different processing times $p_1, p_2, \ldots, p_n$ and distinct due dates $d_1, d_2, \ldots, d_n$ have to be scheduled without preemption. Jobs are delivered in batches. Each batch is delivered after completing all of its jobs. The earliness and tardiness penalty cost for each job would be the same and equal to α and β, respectively.

Traditional JIT models, which consider only earliness and tardiness penalties, have been issued in many papers. One of the first studies on earliness and tardiness problems with distinct due dates was done by

Seidman et al. [3]. They considered due dates as decision variables and solved the problem in polynomial time. Ow and Morton [4], Chand and Chhajed [5], and Baker [6] also investigated different cases of early-tardy problems with distinct due dates. Hendel and Sourd [7] focused on discovering the optimal schedule for earliness-tardiness problems when the job sequence is fixed. This problem is called the timing problem.

Some problems contain a factor that makes them susceptible to delivering jobs in batches. Batching jobs may lead to reducing the cost or time of scheduling, which has attracted particular attention, especially in recent decades [8]. Researchers have generated different problems based on the reasons for batching jobs. Crauwels et al. [9] and Potts and Kovalyov [8] considered batching jobs on machines that require setups when they process jobs of different families. The batch processing machine problem is another kind of batching problem. Lee et al. [10], Li and Lee [11], and Ikura and Gimple [12] were almost the first ones to study this group of batching problems. These machines are capable of working on several jobs simultaneously.

Nowadays, with respect to the growth in e-business competitions and the importance of logistics' role in this field, a new concept in JIT systems evolved that considers the transportation cost according to the number of batches regardless of their sizes. The conjunction of transportation and delivery costs is a new concept that has been investigated in modern JIT approaches [1].

Batch delivery scheduling problems have been studied by Cheng and Kahlbacher [13] for the first time. They worked on earliness penalties and batch delivery costs. This problem was solved in its general form by Cheng and Gordon [14] with processing times related to its batch. They obtained two pseudo-polynomial time solutions for a special case with a fixed upper bound for a number of batches. They proved that the problem is NP-hard and solved its special cases in polynomial time.

Hall and Potts [15] considered a kind of batch delivery problem with transportation constraints on single and parallel machines. They investigated several scheduling problems and provided efficient algorithms for different cases of the problem, such as minimizing tardiness, batch delivery cost, and total completion times.

On the other hand, the adaptability of batching problems makes them suitable for cooperation with other production plan systems. As an example, a combination of batching problems and supply chain management is an important and interesting topic that has attracted much attention. Pundoor et al. [16] studied a system with one supplier and one or more customers. Jobs are delivered in batches, and all jobs that compose a batch belong to one customer. Hall and Potts [17] considered a system in which suppliers work with manufacturers in relation to customers and deliver jobs to them. They considered different versions of batching problems, studied classical scheduling models, and presented an efficient dynamic programming algorithm for their problem. Selvarajah and Steiner [18] discussed a system with one supplier that manufactures multiple products and delivers them to customers in batches. They designed a polynomial time algorithm to minimize holding costs and batch delivery costs from the supplier's point of view. Studying batch delivery problems with the JIT approach has been extended in recent years by Shabtay [19]. In this problem, due dates are controllable, and the objective is to minimize earliness-tardiness, due date assignment, holding, and delivery costs with respect to lead time constraints.

Benmansour et al. [20] proposed two mixed integer programming to minimize the weighted sum of maximum earliness and tardiness costs on a single machine. Rostami et al. [21] developed a mixed integer model to minimize the maximum tardiness and delivery costs with batch delivery and job. They proposed a Branch and Bound (B&B) algorithm to solve the model. Li et al. [22] developed a mathematical model for scheduling the problem of minimizing earliness–tardiness on a single batch processing machine. Ahmadizar and Farhadi [23] presented a mathematical model for a single-machine scheduling problem. In their model, it is supposed that jobs are released at different points in time but delivered to customers in batches. They developed a solution using the ICA.

The main conclusion to be drawn from this review is that earliness-tardiness scheduling problems are practical areas in industrial applications, especially when batching criteria are considered. In spite of their usefulness, batching problems are complicated ones with irregular cost functions. This complexity has been investigated in many papers [24–26]. Garey et al. [27] proved that the single machine earliness-tardiness scheduling problem is NP-hard even for $a_i=b_i=1$ ($i\in N$), where a_i and b_i are earliness and tardiness weights, respectively. From that, it can be easily concluded that our penalty function, with an additional batching consideration, is definitely NP-hard.

The complexity of non-convex objective functions and the time-saving feature of heuristic algorithms allow researchers to investigate meta-heuristic algorithms in order to solve problems more efficiently and with higher quality. For example, Wan et al. [28] used the Tabu search algorithm together with the optimal timing algorithm to solve distinct due window problems. Chang et al. [29] considered a Genetic Algorithm (GA) with artificial chromosomes for minimizing the total deviations on a single machine. They showed that the proposed procedure influences speeding up the convergence of the algorithm.

Allaoua and Algeria [30] studied a common due date earliness-tardiness scheduling problem. They developed a new GA procedure inspired by a dynamic programming algorithm in which the chromosomes and population length are changeable. Lin et al. [31] discussed the same objective function as [32]. They combined the GA and Simulated Annealing (SA) methods to achieve a greedy local search approach to solve their common due date scheduling problem more efficiently.

As a result, in this paper, we address a new version of earliness-tardiness scheduling problems with batch delivery consideration and distinct due dates for the first time, which covers all important concepts that were explained earlier. Due to the limitations imposed by the large size complexity of the proposed model and its nonlinear nature, we develop a meta-heuristic solution approach by using LCA.

It should be emphasized here that the LCA algorithm is used widely in solving complicated real-world problems. Alimoradi and Husseinzadeh Kashan [33] adapted the LCA algorithm for the stock trading rule extraction process. Their approach can be used to extract and save different stock trading rules for different kinds of stock market situations. Bouchekara et al. [34] applied the LCA algorithm for solving the optimal power flow problem. They used this approach to the Algerian power system network for various objectives. Moreover, the LCA has been used in cloud computing systems [35].

3 | Developed Mathematical Model

This section develops a new mixed integer nonlinear mathematical model to formulate earliness-tardiness scheduling problems with batch delivery consideration. Assume a set of n independent jobs scheduling on a single machine. All jobs are available at the starting time of the first job (time zero), and the machine can process only one job at a time. It is assumed to be continuously available without any breakdown or failure.

A set of available jobs with known processing times (π_j) and distinct due dates (d_j) is processed on a single machine and delivered in batches of arbitrary sizes to the customer. A delivery cost (D) would be considered for each batch regardless of its size. Despite processing times and due dates, earliness (α) and tardiness weights (β) are general parameters that are the same for all jobs.

In continuation, we first introduce some notations for our model and then represent our new mathematical model for minimizing the non-convex sum of earliness-tardiness and delivery costs criteria.

Notations

j: index for jobs.

i: index for batches.

n: number of jobs.

d_j: due date of job j.

π_j: processing time of job j.

α: earliness penalty for each early job.

β: tardiness penalty for each tardy job.

D: delivery cost of each batch.

σ: number of batches.

C_i^*: completion time of batch i.

C_j: completion time of job j.

E_j: earliness of job j.

T_j: tardiness of job j.

n_i: number of jobs in batch i.

π_i^*: total processing times of batch i.

All of the parameters in this model are integers.

Decision variables

$$B_i = \begin{Bmatrix} 1, & \text{If } n_i > 0 \\ 0, & \text{Otherwise} \end{Bmatrix}.$$

$$X_{ij} = \begin{Bmatrix} 1, & \text{If job j is assigned to batch i} \\ 0, & \text{Otherwise} \end{Bmatrix}.$$

Now, we can model the objective function and constraints of our nonlinear model like this:

$$\text{Min} \sum_{j=1}^{n} (\alpha E_j + \beta T_j) + D \sum_{i=1}^{n} B_i,$$

Where

$$\sum_{i=1}^{n} X_{ij} = 1, \quad j=1, \dots, n. \qquad (1)$$

$$\pi_i^* = \sum_{j=1}^{n} X_{ij} \times \pi_j, \quad i=1,..,n. \qquad (2)$$

$$n_i = \sum_{j=1}^{n} X_{ij}, \quad i=1,..,n. \qquad (3)$$

$$C_i^* = C_{i-1}^* + \pi_i^*, \quad i=1,..,n, \quad C_0^*=0. \qquad (4)$$

$$C_j = \sum_{i=1}^{n} X_{ij} \times C_i^*, \quad j=1, \dots, n. \qquad (5)$$

$$E_j = \max(d_j - C_j, 0), \quad j=1,\dots,n. \qquad (6)$$

$$T_j = \max(C_j - d_j, 0), \quad j=1,\dots,n. \qquad (7)$$

In this model, the objective function calculates total penalties for weighted earliness, tardiness, and batch delivery costs simultaneously. *Constraint (1)* illustrates that each job should be assigned exactly to one batch. *Constraint (2)* shows that the processing time of each batch is equal to the total processing times of all jobs of the batch. Number of jobs in each batch is presented in *Constraint (3)*. According to *Constraint (4)* and *Constraint (5)*, the completion time of each batch is equal to the completion time of the last job in the batch, and the completion time of each job is equal to the completion time of its related batch, respectively. E_j and T_j variables are the difference between the due date and completion time of job j, as mentioned in *Constraint (6)* and *Constraint (7)*, respectively.

As explained before, the developed model cannot be solved exactly, so the next section provides an LCA algorithm to estimate the optimal solution of the model.

4 | The Proposed LCA

This section develops a LCA algorithm to solve the proposed model in the previous section. First, we introduce the LCA Algorithm and then extend it to solve our developed model.

4.1 | lCA in General

Inspired by the competition of sports teams in an artificial sports league, a new meta-heuristic algorithm called the LCA was introduced by Husseinzadeh Kashan [32]. This algorithm is constructed based on the championship concept and happens in an artificial league space. A sports league is an organization that exists to provide a regulated competition for a number of people to compete in a specific sport. This word is generally used to refer to competitions involving team sports, not individual sports [32]. These teams play with each other under a certain schedule, and the results would be determined as "win-loss or tie". Each team has its own playing style, which helps it achieve its desired result. This formation is usually obtained based on the strengths and weaknesses of players discovered by the coach according to the results of previous contests, which clarifies the team's playing strength and changes needed in the structure and formation of the team. So, it would be changed based on the opponent's playing style and the coach's suggestions for the team in each game. Based on players' abilities and their playing styles, usually, the best team formation would be presented for each team.

To win the game, each team needs to obtain its appropriate plan and formation in every competition. Hence, in the first step, coaches have to analyze their teams and their opponents' playing styles after each match. As an internal policy, this helps them to focus on the strengths and weaknesses of their teams and try to ameliorate their weaknesses and consolidate their strengths. As an external policy, the coach could focus on the weaknesses of the opponent as the opportunities and resist its strengths as the threats to the team. This kind of match analysis is typically known as Strengths/Weaknesses/Opportunities/Threats (SWOT) analysis, which explicitly links internal (strengths and weaknesses) and external factors (opportunities and threats) [36].

After analyzing the games, coaches try to implement the final changes in teams' settings and formations to achieve the best results.

The mentioned structure for a league competition would be implemented in LCA as follows: Like other evolutionary algorithms, LCA starts with an initial population (league size) of individuals called "team". Each team has a particular formation that indicates a feasible solution to the problem. This formation is shown by an n-bit-array in which each bit defines a team member, and any change in the value of this variable may affect its playing style [36]. These teams compete in an artificial league based on the league schedule for several weeks (iterations). They play in pairs, and the results (win or loss) are obtained based on the teams' playing strength (fitness value). With respect to the results obtained by teams and their opponents each week, a new arrangement would be suggested for their formations (new solutions) for the next week. The championship goes on to remain no more weeks (iterations) -stopping condition.

Before explaining the algorithm in detailed form, some important idealized rules that are used to define the basic features of an artificial championship presented by LCA will be presented as follows [36]:

I. It is more likely that a team with better playing strength wins the game. The term "playing strength" refers to a team's ability to beat another team.

II. The outcome of the game is not predictable, given the known playing strength of the teams perfectly. In other words, it is not unlikely that FC Barcelona loss the game to Sandoghe_Zakhire_Robat_Karim from the Iranian 3rd division.

III. The probability that team i beats team j is assumed to be equal from both teams' points of view.

IV. The outcome of the game is only a win or loss; there is no tie.

V. When team i beats team j, any strength that helped team i to win has a dual weakness that caused team j to lose. In other words, any weakness is a lack of a particular strength. An implicit implication of this rule is that although the match outcome may be imputed to chance, technical staff may not believe it.

VI. Teams focus only on their upcoming contest without regard to the other future matches. Formation settings are done based on the events of the previous week.

For more illustrations, the main steps of the algorithm are shown in *Fig. 1* and explained in detail in Sections 4.1.1-4.1.3. Also, for more information, we refer the readers to [37].

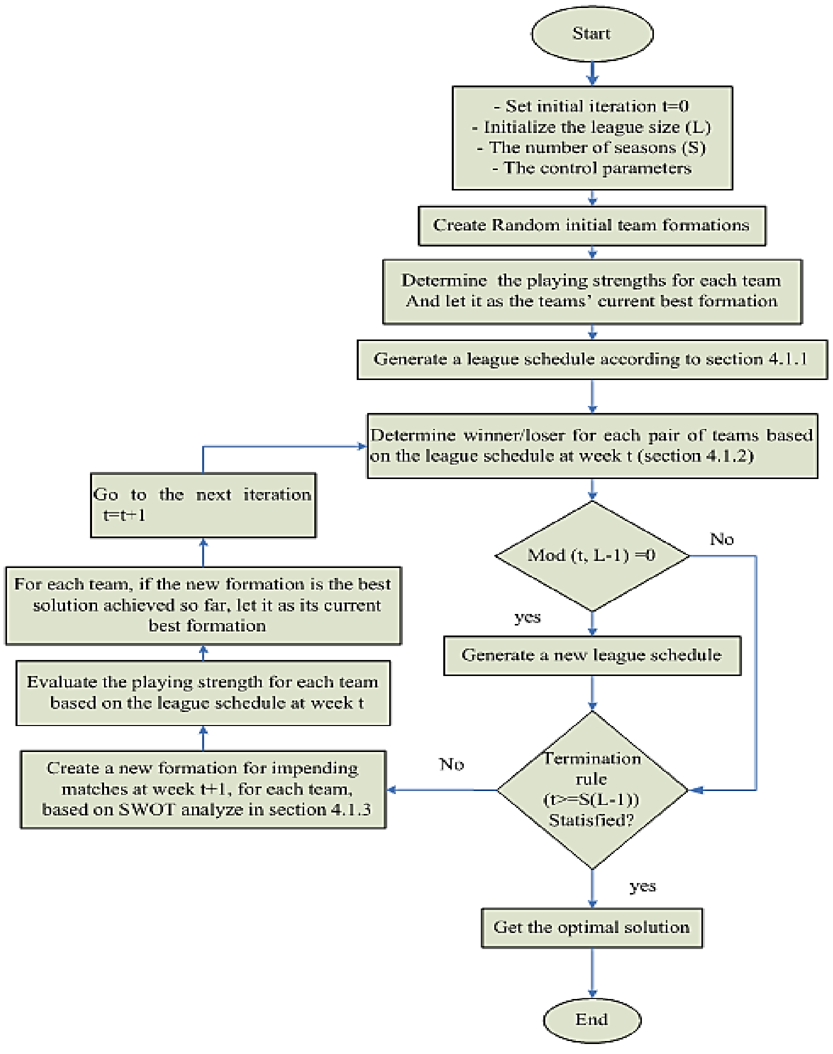
Start
- Set initial iteration t=0
- Initialize the league size (L)
- The number of seasons (S)
- The control parameters
Create Random initial team formations
Determine the playing strengths for each team And let it as the teams' current best formation
Generate a league schedule according to section 4.1.1
Determine winner/loser for each pair of teams based on the league schedule at week t (section 4.1.2)
Go to the next iteration t=t+1
For each team, if the new formation is the best solution achieved so far, let it as its current best formation
Evaluate the playing strength for each team based on the league schedule at week t
Create a new formation for impending matches at week t+1, for each team, based on SWOT analyze in section 4.1.3
Mod (t, L-1) =0
No
yes
Generate a new league schedule
No
Termination rule (t>=S(L-1)) Statisfied?
yes
Get the optimal solution
End

Fig. 1. The framework of LCA in general.

4.1.2 | Determining winner/loser

In real-world league systems, the result of each competence is determined in terms of win, loss, or tie, with 3, 0, and 1 points for each. However, in LCA, there is no tie for teams, and the results are defined only in terms of win or loss. In a stable condition, the more powerful teams have a better chance of winning than the weaker ones. This means that the chance of each team winning is proportional to its playing strength (idealized rules 1 and 2).

In LCA, the winner and loser are determined randomly based on the value of the fitness function for each team. This value is proportional to the team's playing strength and is measured according to its distance with an ideal function value ($\hat{f}$). This ideal function could be estimated by the minimum value of $\hat{f}^t$ found so far. This function is calculated based on the best formation of each team (*Eq. (8)*), such as i, obtained till week t, $B_i^t = (b_{i1}^t, b_{i2}^t, \ldots, b_{in}^t)$:

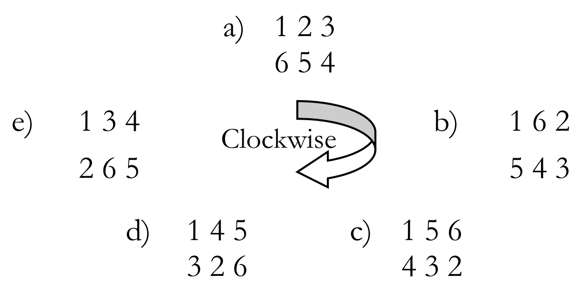

Fig. 2. An example of a round-robin scheduling algorithm.

$$\hat{f}^{t} = \min\{f(B_i^t)\}. \tag{8}$$

$$\frac{f(X_i^t)-\hat{f}}{f(X_j^t)-\hat{f}} = \frac{p_j^t}{p_i^t}. \tag{9}$$

In *Eq. (9)*, $X_i^t = (x_{i1}^t, x_{i2}^t, ..., x_{in}^t)$ and $X_j^t = (x_{j1}^t, x_{j2}^t, ..., x_{jn}^t)$ are the opponents" formations, team i and j, at week t. $f(X_i^t)$ and $f(X_j^t)$ show the playing strength (objective value) relevant to X_i^t and X_j^t, respectively.

Due to idealized rule 3, the probability that team i beats team j at week t (p_i^t) is assumed to be equal to the state that team j beats i (P_j^t). So, we have:

$$p_i^t + p_j^t = 1. \tag{10}$$

From 3 and 4, we can write:

$$p_i^t = \frac{f(X_j^t)-\hat{f}}{f(X_i^t)+f(X_j^t)-2\hat{f}}. \tag{11}$$

To assign a win or loss to each team based on the second and fourth idealized rules, a random number is generated in [0, 1]. If the value is less than or equal to p_i^t, means that team i beats j; otherwise, j wins the match. If $f(X_i^t)$ be very close to $f(X_j^t)$, then $p_i^t \rightarrow 1/2$, and if $f(X_j^t) >> f(X_i^t)$, then $p_i^t \rightarrow 1$. These cases are evidence for our winner/loser approach validation [36].

4.1.3 | Setting up a new team formation

One of the most important parts of LCA implementation is updating the team formation in order to achieve new solutions. A coach needs to recognize the strengths and weaknesses of the individual members and also the opportunities and threats of the own team against its opponents in order to arrange the best formation consistent with each member's ability and the team as a whole. Similar to real sports teams, in LCA, after each competition, the coach analyzes the team's playing style (as an internal strategy) and also its post-match opponent's contest in the current week (as an external strategy) to arrange the team formation for the next week. According to the different statuses that take place for each team when they lose or win, the SWOT matrix is designed. In this matrix, any win (loss) results for each team are interpreted as the direct effect of the team's strength (weakness) as an internal factor or the opponent's weakness (strength) as an external factor. For example, assume that the next match of team i will be against team l in week t+1. If team l wins the game in week t, then this success might be a direct threat to team i. On the other hand, assume that team i also won its own game in week t. So, it is reasonable that team i focuses on its strengths to overcome the threats of team l and be the winner of the game in week t+1.

Table 1. Decision-making matrix from the team coach's point of view and based on the win or loss status of the teams [28].

	I: Winner **L: Winner**	**I: Winner** **L: Loser**	**I: Loser** **L: Winner**	**I: Loser** **L: Loser**
S	Own strengths (or weaknesses of j)	Own strengths (or weaknesses of j)	-	-
W	-	-	Own weaknesses (or strengths of j)	Own weaknesses (or strengths of j)
O	-	Weaknesses of l (or Strengths of k)	-	Weaknesses of l (or Strengths of k)
T	Strengths of l (or Weaknesses of k)	-	Strengths of l (or Weaknesses of k)	-

As shown in the SWOT matrix (*Table 1*), this strategy is called the S/T strategy, which focuses on the strength of team i because of its victory at week t and the threats from team l that must be conquered by team i.

This matrix helps the coach analyze the teams and fill the gaps simultaneously. As for internal reasons, gaps might be the result of bad formation of the team, bad positioning, or poor performance of members or the whole team. Also, external characteristics such as the opponent's strength or weakness and playing style might be other important factors in determining the gaps.

Based on the strategic matrix and with respect to the coach's improving analysis to fill the gaps, the new formation X_i^{t+1} for team i at weak t+1 would be determined by one of the following equations. Note that these equations are developed with the assumption that teams play with their current best formations, which found them suitable till now.

Based on S/T strategy:

$$x_{id}^{t+1} = b_{id}^{t} + y_{id}^{t}(c_1 r_1 (x_{id}^{t} - x_{kd}^{t}) + c_1 r_2 (x_{id}^{t} - x_{jd}^{t})), \quad \text{for all } d = 1,\ldots,n. \tag{12}$$

Based on S/O strategy:

$$x_{id}^{t+1} = b_{id}^{t} + y_{id}^{t}(c_2 r_1 (x_{kd}^{t} - x_{id}^{t}) + c_1 r_2 (x_{id}^{t} - x_{jd}^{t})), \quad \text{for all } d = 1,\ldots,n. \tag{13}$$

Based on W/T strategy:

$$x_{id}^{t+1} = b_{id}^{t} + y_{id}^{t}(c_1 r_2 (x_{id}^{t} - x_{kd}^{t}) + c_2 r_1 (x_{jd}^{t} - x_{id}^{t})), \quad \text{for all } d = 1,\ldots,n. \tag{14}$$

Based on W/O strategy:

$$x_{id}^{t+1} = b_{id}^{t} + y_{id}^{t}(c_2 r_2 (x_{kd}^{t} - x_{id}^{t}) + c_2 r_1 (x_{jd}^{t} - x_{id}^{t})), \quad \text{for all } d = 1,\ldots,n. \tag{15}$$

In the above formulas, j and k represent the index of the opponent of "team i at week t+1", "team i at week t," and "team l at week t," respectively. r_1 and r_2 are uniformly distributed in [0,1], c_1 and c_2 are constant coefficients in relation to the contribution of the strength and weakness components, respectively, and d = 1,..., n is the dimension index. To move toward the winner or abstain from the loser, the + or – signs were devised in parenthesis. y_{id}^{t} is a binary variable called the change variable. The value 1 means that b_{id}^{t} would be changed, and 0 means that the change is not necessary. As these changes often occur in some dimensions, we could randomly select q_i^t dimensions from B_i^t and assign the value 1 to them.

A truncated geometric distribution was used to set the rate of changes dynamically. It could be determined as follows:

$$q_i^t = \left\lceil \frac{\ln(1-(1-(1-p_c)^{n-q_0+1})r)}{\ln(1-p_c)} \right\rceil + q_0 - 1 : \; q_i^t \in \{q_0, q_0+1, \ldots, n\}. \tag{16}$$

In this formula, *r* is a random number in [0, 1], and p_c is the probability of success in the truncated geometric distribution. There are two possible ways to analyze and arrange the team formations based on the SWOT strategy in the first procedure, which is presented by *Eqs. (12)-(15)*, teams' recent formations are used to generate the new formations at week t+1 (Type of formation 1), and in the second procedure, *Eqs. (17)-(20)*, analysis of the post-match is done based on the team's best formation (b^t) instead of its previous formation (X^t) (Type of formation 2). Then we have:

S/T:

$$q_i^t = \left\lceil \frac{\ln(1-(1-(1-p_c)^{n-q_0+1})r)}{\ln(1-p_c)} \right\rceil + q_0 - 1:\ q_i^t \in \{q_0, q_0+1, \ldots, n\}. \tag{17}$$

S/O:

$$q_i^t = \left\lceil \frac{\ln(1-(1-(1-p_c)^{n-q_0+1})r)}{\ln(1-p_c)} \right\rceil + q_0 - 1:\ q_i^t \in \{q_0, q_0+1, \ldots, n\}. \tag{18}$$

W/T:

$$q_i^t = \left\lceil \frac{\ln(1-(1-(1-p_c)^{n-q_0+1})r)}{\ln(1-p_c)} \right\rceil + q_0 - 1:\ q_i^t \in \{q_0, q_0+1, \ldots, n\}. \tag{19}$$

W/O:

$$q_i^t = \left\lceil \frac{\ln(1-(1-(1-p_c)^{n-q_0+1})r)}{\ln(1-p_c)} \right\rceil + q_0 - 1:\ q_i^t \in \{q_0, q_0+1, \ldots, n\}. \tag{20}$$

4.2 | Proposed LCA Approach to JIT Batch Delivery Problem

The LCA is a new meta-heuristic algorithm that solves continuous optimization problems. We used this algorithm to optimize our discrete non-convex earliness-tardiness cost function with batch delivery criteria. Adopting the algorithm with our discrete model, several representational schemes were added to the algorithm. Some important features, such as discrete individuals instead of continuous ones and redundancy elimination in our population, which leads to better solutions closer to optimal ones and less computational time, were devised in our proposed LCA. Following, we explain the main steps of the proposed LCA.

4.2.1 | The representation scheme

One of the important parts of meta-heuristic algorithms is the representation scheme. In our proposed LCA, this procedure helps us to show the solution characteristics in the form of teams called "team's playing formation." This formation is presented by $X_i^t = (x_{i1}^t, x_{i2}^t, \ldots, x_{in}^t)$, which is defined as ith team's formation (solution) at week (iteration) t. x_{ij}^t shows that the jth job of solution i is placed in $x_{ij}^{t\,(th)}$ batch. We use several representational schemes to map the solutions generated by LCA to our model's discrete space.

The first scheme, which is used to represent solutions in discrete models, is implementing the algorithm in continuous space and generating solutions (X_i^t) in the form of continuous variables, $x_{ij}^t \in [1, n]$, where n is the number of jobs to be scheduled. In order to define sequences and fitness values for each solution, at the end of each iteration, variables (our teams) will be converted to discrete ones by rounding them up (LCA-1).

In contrast with the former procedure, in the second scheme, our initial population (teams) and all new team formations are constructed based on discrete variables, $x_{ij}^t \in \{1,2,\ldots,n\}$, where n is the number of jobs to be scheduled. In this scheme, all the solutions used in the whole body of our algorithm are integers (LCA-2). The difference between the two procedures is that in the first procedure, our solutions compete in continuous space; in order to calculate the objective function value and compare team formations (sequences), at the end of each iteration, they convert to integer ones. However, in the second procedure, discrete variable solutions compete in a discrete league with discrete formations, and finally, solutions are returned in discrete forms. Comparison results (shown in *Table 3*) prove that the latter procedure performs much better than the former one, and it would be better to use the second procedure for generating our population. Based on this procedure, our solution presents a vector whose length is equal to the number of jobs (n).

Each bit in this array shows the batch ID that could be an integer from 1 to n. For example, a sample solution obtains as follows: A=[1 3 4 2], which means the first job is assigned to batch 1, the second job to batch 3, the third job to batch 4, and the fourth one to batch 2.

4.2.2 | The initial population

Generating a set of initial solutions is the first step to implementing the LCA. There are three major steps in our proposed LCA accomplishment. The first step is to determine the appropriate population size, which is not too small and not too large, in order to achieve appropriate solutions in a reasonable time. The second step is generating a random initial population containing primal random sequences. The quantity of the primal sequences is equal to the population size determined in the first step [38]. As can be seen in *Table 3* (columns 1 and 2), the proposed LCA-2 seems to be appropriate for solving our model. However, repetitive sequences in each population indicate that this algorithm suffers from redundancies when generating solutions. For example, the solution [1 2 2 3 4] is equal to [2 3 3 4 5], [1 3 3 4 5], and some other ones, which may lead the algorithm to trap local optimums. Hence, it would be helpful to eliminate redundancies in initial populations in the third step (LCA-3). In order to apply this decision to our proposed LCA, first, we should standardize the solutions generated in Step 2 following this procedure:

Step 1. i=1.

Step 2. Find the ith minimum batch IDs in the solution array and assign them number i.

Step 3. i=i+1, and go to Step 2.

Step 4. Repeat Steps 2 and 3 till all jobs are sorted.

The illustration example for standardization is presented in *Fig. 3*. In Step (a), the minimum batch IDs 2), in this example) are found and replaced with the number i=1. In the second step (Step (b)), the second minimum batch IDs in the new solution array 3), in this example, are found and replaced with number 2. This procedure is repeated until Step (d), in which all jobs are sorted from 1 to 4.

An appropriate algorithm is developed in order to generate random and non-repetitive solutions. This is done by eliminating redundancies in our population. As shown in *Fig. 4*, an initial random solution is first generated and standardized based on the standardization procedure (T_0). Then, in each step, new solutions (T_i) are generated, standardized, and compared with the former ones (T_0, T_1, ..., T_{i-1}). If T_i is equal to each of (T_0, T_1, ..., T_{i-1}), it will be replaced with a newly generated solution called (T_i). The algorithm terminates when the desired number of population members is achieved.

Original state	3	4	3	2	2	6
Step 1	3	4	3	1	1	6
Step 2	2	4	2	1	1	6
Step 3	2	3	2	1	1	6
Step 4	2	3	2	1	1	4

Fig. 3. An example of solution standardization.

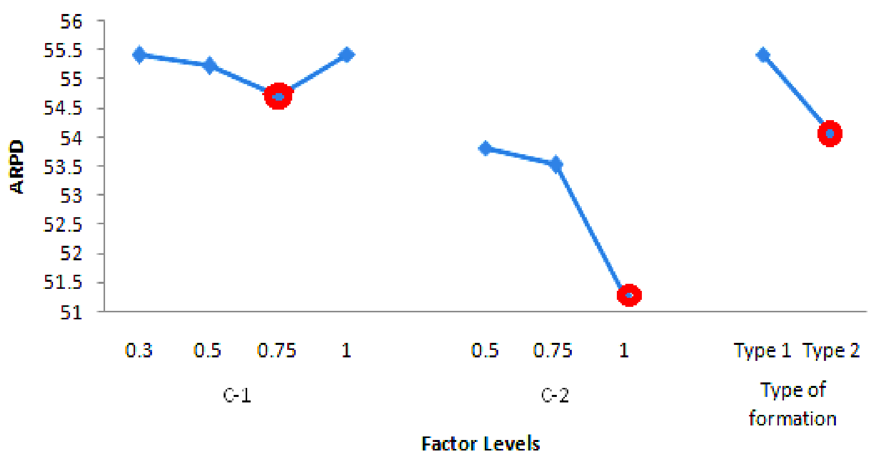

Fig. 4. The ARPD results for control parameters of LCA.

In order to test the output of LCA under each mapping scenario, a comparison is done between the proposed scenarios. For this purpose, the evaluation is done by using the Average Relative Percentage Deviation (ARPD) method defined by Ruiz et al. [39].

$$\text{ARPD} = \sum_{i=1}^{R} \left(\frac{(S_i - S_{best}) \times 100}{S_{best}}\right) \Big/ R. \tag{21}$$

In this equation, R is the number of replications for each set. S_i is the solution obtained by the proposed algorithm, and S_{best} is the best result during the R replications. 48 different sets of test problems, each containing 10 instances, were used to testify to the performance of our scenarios. The name of the benchmark problems is demonstrated in *Table 2*.

From *Eq. (21)*, it is clear that the smaller the value of ARPD, the better the algorithm's performance. Due to this fact, the average ARPD results demonstrated in *Table 3* are symptomatic of a highly significant improvement in LCA-2 and LCA-3 in comparison with LCA-1. Also, drawing a parallel between LCA-2 and LCA-3 strongly proves that using the latter procedure gives rise to achieving better solutions.

4.2.3 | Fitness evaluation

The fitness evaluation aims to calculate the goodness of the candidate solutions. Owing to our objective function, the teams' fitness values are obtained by calculating jobs' weighted earliness- tardiness and batch delivery costs. In LCA, the winner/ loser is determined randomly based on the value of the fitness function for each team. This value is proportional to the team's playing strength and is measured according to its distance with an ideal function value ($\hat{f}$). Since this ideal function is not in hand, it could be estimated by the minimum value of $\hat{f}^t$ found so far (See *Eq. (8)*).

Table 2. Name of problems generated with variable parameters.

N	**Delivery Cost=Rand(100,4000)**				**Delivery Cost=Rand(4001,12000)**				**Delivery Cost=Rand(12001,30000)**			
	T=0.3		**T =0.6**		**T =0.3**		**T=0.6**		**T =0.3**		**T =0.6**	
	ϱ=0.5	ϱ=1	ϱ=0.5	ϱ=1	ϱ=0.5	ϱ=1	ϱ=0.5	ϱ=1	ϱ=0.5	ϱ=1	ϱ=0.5	ϱ=1
Code	ETBD1	ETBD2	ETBD3	ETBD4	ETBD5	ETBD6	ETBD7	ETBD8	ETBD9	ETBD10	ETBD11	ETBD12

Table 3. Comparison between different versions of LCA based on ARPD test values.

Code	Num of Jobs	LCA-1 ARPD-1	LCA-2 ARPD-2	LCA-3 ARPD-3
ETBDS1	10	9.650883	0	0.385471
ETBDS2	10	0.576727	0.056075	0
ETBDS3	10	0	0	0.002005
ETBDS4	10	0.040008	0	0
ETBDS5	10	10.49263	0	0
ETBDS6	10	6.992513	0	0
ETBDS7	10	5.911086	0	0
ETBDS8	10	8.539073	0	5.628873
ETBDS9	10	37.00386	16.99365	0
ETBDS10	10	50.9369	5.303566	11.49748
ETBDS11	10	43.5099	0	10.01565
ETBDS12	10	10.15265	10.15265	0
ETBDS1	50	79.18064	1.850438	0.832094
ETBDS2	50	57.21937	2.021437	1.736641
ETBDS3	50	15.03564	1.028389	0.473815
ETBDS4	50	20.3549	3.203455	0
ETBDS5	50	117.9944	0.892572	2.701348
ETBDS6	50	95.68398	3.088849	2.03969
ETBDS7	50	29.79702	0.57392	1.302373
ETBDS8	50	27.12109	1.034036	1.674764
ETBDS9	50	134.1525	7.9931	0.665525
ETBDS10	50	98.8927	1.583097	2.485106
ETBDS11	50	37.8745	2.79131	0.07922
ETBDS12	50	51.28559	0	3.630047
ETBDS1	100	-	16.06418	3.213133
ETBDS2	100	-	14.17521	34.27443
ETBDS3	100	-	0.547844	0.433605
ETBDS4	100	-	0.615368	1.179192
ETBDS5	100	-	4.279033	2.205131
ETBDS6	100	-	0.828069	2.967736
ETBDS7	100	-	0.156325	0.883283
ETBDS8	100	-	2.277285	0.930146
ETBDS9	100	-	5.463604	0.051317
ETBDS10	100	-	2.122505	0
ETBDS11	100	-	1.69648	0.260135
ETBDS12	100	-	0.927387	0.645785
AVERAGE		39.51661	2.992218	2.560944

5 | Computational Results

5.1 | Data Generation

In this paper, test problems were designed based on data generation schemes developed by Abdul-Razaq and Potts [40] and Ow and Morton [4]. Processing times, earliness, and tardiness weights were chosen to be randomly generated integers inside [10,100], [1, 10], and [1, 10], respectively. For the due dates, we considered T (the tardiness factor) and ϱ (the due date range) in sets {0.3, 0.7} and {0.6, 1.2}, respectively. Our due dates were generated uniformly over the integers $\{\pi.\ (1\text{-}\mathrm{T} - \rho/2),\ldots, \pi.(1\text{-}\mathrm{T} + \rho/2)\}$, where π is the total processing times of all jobs. The probable interactions between batch delivery costs and earliness-tardiness penalties may lead to increased problem hardness. To consider special cases, different intervals were presented for delivery costs. The intervals were generated randomly over integers inside [100, 4000], [4001, 12000], and [12001, 30000].

5.2 | Parameters Tuning

Parameter setting is an important part of designing appropriate solutions. In this section, three important parameters of our proposed algorithm were determined and tuned. With a view to doing this, three different groups of jobs n={10, 50, 100}, each containing a set of 12 instances, generated and replicated 5 times. Generally, 180 test problems were used for tuning the parameters.

As can be seen in *Table 4*, the first and second parameters, c_1 and c_2, are constant coefficients used to design teams' new formations for the next week. These parameters are in relation to the contribution of the strength and weakness components, respectively. The third parameter is the "Type of formation". As explained in Section 4.1.3, there are two alternative procedures for arranging a new team formation. The first one uses the recent formation of teams (Type of Formation 1), and the second one uses the best formation of teams (Type of Formation 2).

To evaluate the performance of solutions obtained by different levels of decision factors, the ARPD method from *Eq. (14)* is used once more.

The following procedure is used to define the best levels for each parameter:

I. Select one of the three parameters (for example, C_1) in order to define its best level.

II. Fix the two other parameters to their first levels.

III. With respect to the ARPD result, set the selected parameter's level to the value with minimum ARPD (third level for C_1=0.75).

IV. Select the other parameter (like C_2).

V. Set C_1 to its best level and Type of formation to its first level.

VI. Same as the third step, set C_2 to its best (C_2=1).

VII. A similar procedure will be used by fixing C_1 and C_2 to their best levels and determining the third parameter's best level.

The results of our tests depicted in *Fig. 5* demonstrate ARPD results for different levels of C_1, C_2, and Types of formation. As clearly revealed in this figure, levels {3, 3, 2} with minimum ARPD value are the best levels for C_1, C_2, and Type of formation, respectively.

Table 4. Factors and factor levels for parameter tuning.

Factors	Level
c_1	0.3 - 0.5 - 0.75 - 1
c_2	0.5 - 0.75 - 1
Type of formation	Type 1 - Type 2

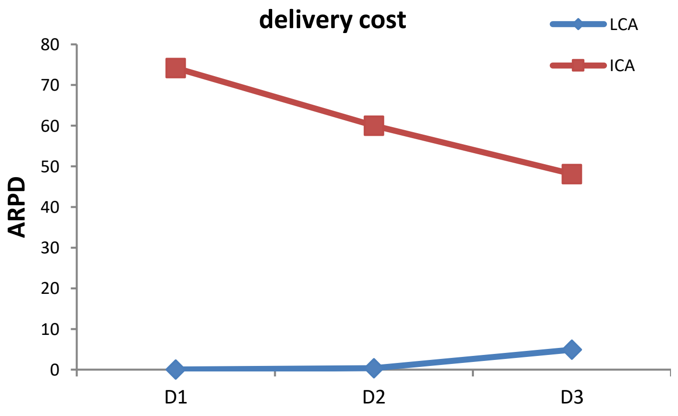

Fig. 5. Overall ARPD comparison between LCA and ICA based on different sets of delivery costs.

5.3 | Experimental Results

Two different schemes are considered when evaluating the performance of the proposed LCA. A comparison between the results of the proposed LCA and the mathematical modeling approach is made as the first scheme. Test problems are generated randomly in 5 sets of $n = \{6, 9, 10, 15, 18\}$, each with 12 subsets and 10 replications. As can be seen in *Table 5*, in comparison with global solutions obtained by Lingo, our proposed algorithm reaches global optimums in most runs (Column 2). The average gap between LCA and Lingo, which is given in Column 3 and calculated by *Eq. (22)*, reveals that the difference between global solutions and LCA's best solutions is very small and negligible. This result verifies that our algorithm is powerful enough to reach the optimal solutions that are very close to or equal to global ones. For more illustration, some of the Lingo and LCA results, along with their minimum, maximum, and best solutions, are reported in *Table 6*.

$$\mathrm{GAP} = \frac{\left(\mathrm{obj}_{\mathrm{heuristic}} - \mathrm{obj}_{\mathrm{global\ sol}}\right)}{\mathrm{obj}_{\mathrm{global\ sol}}}. \tag{22}$$

Results of ARPD tests in Columns 4 and 5, which are used to compare Lingo's and LCA's best solutions, corroborate that solutions obtained by the proposed LCA generally have great superiority over Lingo's results. Columns 6 and 7, for computational times, validate the high performance of our LCA in comparison to Lingo's prolix CPU times. The results indicate that even for problems with 15 jobs, Lingo is not able to reach global optimums or at least the best solutions equal to LCA after more than 1500 seconds of CPU time. These critical limitations prove that a meta-heuristic is needed to solve the remaining scenarios. Therefore, as the second scheme and for the purpose of measuring the performance of our LCA for large sizes of jobs, another meta-heuristic is applied to our model. The algorithms were run on a Pentium 4 computer with a 1.87 GHz CPU and 1 GB of RAM.

Table 5. ANOVA test for ARPD results of c_1 factor.

Source	DF	SS	MS	F	P
c_1	3	12	4	0	1.00
Error	140	143361	1024		
Total	143	143374			

Table 6. Comparison of the quality of the results obtained by Lingo 9 and the proposed LCA.

Number of Jobs	Global Optimal (%)	GAP (%)	Average Error for Lingo	Average Error for LCA	Average CPU Time for Lingo (sec)	Average CPU Time for LCA (sec)
6	100	0	0.384	0	2.92	5.74
9	0.625	0.26	33.857	0.164217	11.79	6.50
10	0.95833	3.05	0.588	2.924710	15.24	6.41
15	0.59166	1.195	5.980	0.70735	1520.63	7.41
18	0.00833	--	19.03	3.82292	2298.44	8.60

As mentioned above, a population-based meta-heuristic is used to validate the performance of our LCA, especially for large-size problems. This evolutionary algorithm, which is based on social and political relations between imperialists, is called the ICA. To keep the paper short, the readers are referred to the main version of this algorithm, which was introduced by Atashpaz and Lucas [41]. Some changes are devised in ICA to be applicable to our model.

The number of countries in the population space and the maximum iteration number in ICA are set as 20 and 950, respectively. Also, the LCA parameters are set as follows: $L=20$, $S=50$, $c_1= 0.75$, $c_2=1$, $P_c =0.01$.

There are 12 combinations for 5 different sets of jobs, $n=\{10, 20, 50, 100, 200\}$. Each combination is repeated 10 times, and consequently, 600 instances are generated in general. Both algorithms were coded in MATLAB 7.3 and run on the same computer, as mentioned earlier.

ARPD test defined in *Eq. (14)* is still used for comparison. The results of our 600 instance problems are summarized in *Table 6*. The average solutions obtained by 10 replications for LCA and ICA are reported in

Columns 3 and 4, respectively. It is obvious that by increasing the problem size, the solution quality obtained by LCA will dramatically increase compared with ICA. This conclusion is obtained by comparing ARPD values in Columns 5 and 6. For a number of jobs greater than 20, there is not any solution obtained by ICA that dominates a solution by LCA.

In *Fig. 6* and *Fig. 7*, the ARPD results were depicted for LCA and ICA comparison based on different sets of delivery costs, due dates, and number of jobs, respectively. The results demonstrate that our proposed LCA strictly outperforms ICA in all different categories, which may influence the problem hardness, especially for problems with a large number of jobs.

Table 7. Comparison between LCA and ICA results.

Code	Num of Jobs	LCA	ICA	LCA	ICA
		Average of 10 Replication		ARPD	
ETBDS1	10	6933.4	7018.8	0	0.965114
ETBDS2	10	7368.6	7299.3	0.847292	0
ETBDS3	10	9988.5	10111.2	0	1.350801
ETBDS4	10	9087.5	9101.2	0	0.156639
ETBDS5	10	17266.7	17265.1	0.033795	0.016861
ETBDS6	10	18250.6	17913.6	1.822512	0.023172
ETBDS7	10	22449.9	22482.8	0	0.165036
ETBDS8	10	26661.9	26735.9	0	0.476129
ETBDS9	10	17266.7	33187.1	0.392165	112.2046
ETBDS10	10	18250.6	33418.2	0	94.04584
ETBDS11	10	22449.9	44272.8	0	111.5918
ETBDS12	10	26661.9	31937.2	10.75553	36.87526
ETBDS1	20	12833	15629.3	0	20.81877
ETBDS2	20	19061.5	26886.5	0	38.80924
ETBDS3	20	26971.9	30216.4	0	12.62475
ETBDS4	20	25700.7	30311.2	0	18.23471
ETBDS5	20	28687.4	34248.5	0	20.11084
ETBDS6	20	32163.6	34716.4	2.704313	10.32669
ETBDS7	20	49411.8	54295.6	0.245334	9.582612
ETBDS8	20	37294.5	42666.7	0.415775	13.92655
ETBDS9	20	53234.2	47610.5	17.15054	3.183035
ETBDS10	20	73280.2	54286.3	37.21648	0
ETBDS11	20	81543.6	76001.2	10.55761	1.105225
ETBDS12	20	71706.5	68643.9	10.99929	3.364461
ETBDS1	50	55214.8	132196.5	0	141.4118
ETBDS2	50	56552	138815.8	0	171.3781
ETBDS3	50	84383.2	144579.5	0	70.34534
ETBDS4	50	77526.5	162199.8	0	116.7708
ETBDS5	50	82231.9	159678.8	0	92.43738
ETBDS6	50	92711.4	217603.7	0	137.0658
ETBDS7	50	180265.8	278039.8	0	51.86653
ETBDS8	50	141557.2	242018.7	0	80.58503
ETBDS9	50	168704.2	211266.1	2.902811	24.66006
ETBDS10	50	136956	204005.8	4.706995	57.4304
ETBDS11	50	274451.6	322529.7	2.658905	19.84335
ETBDS12	50	241628.5	330295.4	0.857699	36.5041
ETBDS1	100	290958.5	647923	0	128.398
ETBDS2	100	395657.1	828531.2	0	116.1921
ETBDS3	100	371169.1	678805.4	0	89.17491
ETBDS4	100	248646.7	642068.9	0	166.7628
ETBDS5	100	415708.1	896631.2	0	116.1802
ETBDS6	100	509396.9	994780.5	0	102.9088
ETBDS7	100	456470.2	827449.4	0	81.00827
ETBDS8	100	391116.5	956045.5	0	152.2525
ETBDS9	100	648946.8	978028.8	0	51.68802
ETBDS10	100	729042.2	1158114	0	60.82529
ETBDS11	100	752147.2	1026821	0	35.47066
ETBDS12	100	634358.5	1103657	0	76.54836
ETBDS1	200	1453547	2579712	0	86.33171

Table 7. Continued.

Code	Num of Jobs	LCA	ICA	LCA	ICA
		Average of 10 Replication			
ETBDS2	200	2022690	3479661	0	78.16464
ETBDS3	200	1577123	2726755	0	74.87251
ETBDS4	200	1360956	3310455	0	150.8562
ETBDS5	200	1903113	3042460	0	71.14618
ETBDS6	200	2227132	3644104	1.520104	74.50943
ETBDS7	200	1980315	3183278	0	61.46818
ETBDS8	200	1759632	3908530	0	124.8984
ETBDS9	200	2968583	4187307	0	54.36997
ETBDS10	200	2771882	3909955	0	39.1167
ETBDS11	200	2706385	3886286	0	42.8141
ETBDS12	200	2267609	4495491	0	100.9322
Average		552488.2	940205.5	1.763119	60.78578

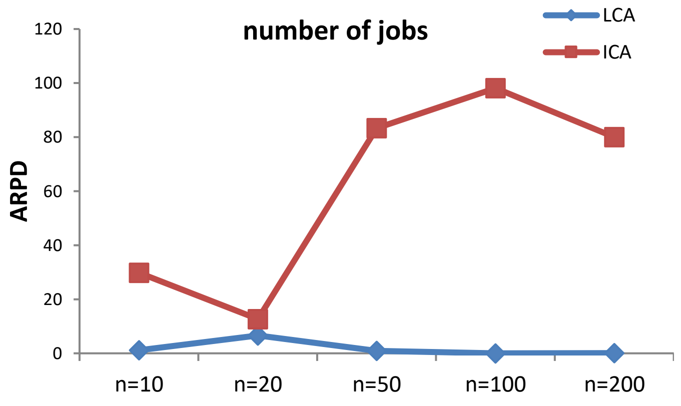

Fig. 6. Overall ARPD comparison between LCA and ICA based on different sets of due dates.

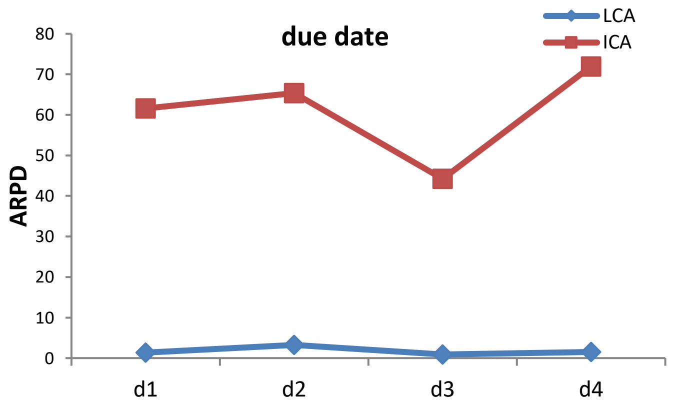

Fig. 7. Overall ARPD comparison between LCA and ICA based on different sets of jobs.

Table 8. T-test for ARPD results of LCA and ICA.

	N	Mean	St Dev	SE Mean	p
LCA	60	1.76	5.66	0. 73	0.00
ICA	60	60.8	50.2	6.5	

In order to determine whether the differences between LCA and ICA solutions are statistically meaningful or not, a t-test is employed on the ARPD results. The hypothesis testing for all test problems is as follows:

Null hypothesis: the average deviation of LCA=the average deviation of ICA.

Alternative hypothesis: the average deviation of LCA<the average deviation of ICA.

As reported in *Table 8*, the computed p-value forICA-LCA's statistical comparison is equal to 0.00, which means that the null hypothesis is strongly rejected. The t-test result substantiates that the difference between the best solutions of the LCA and ICA is strictly significant.

On the other hand, the comparison between mean and standard deviation results in *Table 8* (Columns 3 and 4) significantly reveals major differences between LCA's and ICA's best solutions.

6 | Conclusion

In this paper, we have proposed a new discrete meta-heuristic algorithm based on the LCA to solve the earliness-tardiness batch delivery problem in a one-machine environment with a non-regular cost function. Since LCA works in continuous space, we use several representational schemes to map the solutions generated by LCA to discrete space and compare the output of the algorithm under each mapping scenario. To increase the quality of our proposed LCA, some important parameters have been selected and tuned using ARPD evaluation tests.

Finally, two different comparison schemes based on a mathematical modeling approach and heuristic methods were developed and implemented by using a lingo system and a discrete version of the ICA, respectively. The Lingo results confirm the validity of our proposed LCA, both in time and in terms of solution quality. The comparison results verified that the high complexity of our model affected Lingo results and caused the algorithm not to be suitable even for small sizes of jobs (more than 15). Applying the ICA as a population-based meta-heuristic on our model strongly verified that our LCA is a state-of-the-art algorithm to solve our problem in a reasonable time and with appropriate solution quality. The encouraging results give hope to authors in applying the proposed LCA to other combinatorial problems in the near future.

Acknowledgment

The authors are grateful to the introducer of LCA, Ali Husseinzade Kashan, for his support and help in bringing out this paper.

Author Contributions

Zahra Pourali developed the mathematical model, designed the experiment, and drafted the manuscript. Bohlool Ebrahimi contributed to the implementation of the League Championship Algorithm (LCA), data analysis, and manuscript revisions. Both authors collaborated in interpreting the results and finalizing the manuscript.

Funding

This research did not receive any specific funding from public, commercial, or not-for-profit entities.

Data Availability

The data supporting the study’s findings are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Ongsakul, V. (2001). *Common due date scheduling with batch delivery* [Thesis]. https://ttu-ir.tdl.org/server/api/core/bitstreams/1cd3e7ae-46c9-42eb-b649-e1d37ab517e4/content

[2] Husseinzadeh, K. A. (2014). League championship algorithm (LCA): an algorithm for global optimization inspired by sport championships. *Applied soft computing journal*, *16*, 171–200. DOI: 10.1016/j.asoc.2013.12.005

[3] Seidmann, A., Panwalkar, S. S., & Smith, M. L. (1981). Optimal assignment of due-dates for a single processor scheduling problem. *International journal of production research*, *19*(4), 393–399. DOI: 10.1080/00207548108956667

[4] Ow, P. S., & Morton, T. E. (1989). The single machine early/tardy problem. *Management Science*, *35*(2), 177–191. DOI: 10.1287/mnsc.35.2.177

[5] Chand, S., & Chhajed, D. (1992). A single machine model for determination of optimal due dates and sequence. *Operations research*, *40*(3), 596–602. DOI: 10.1287/opre.40.3.596

[6] Baker, K. R. (1998). *Elements of Sequencing and Scheduling, Hanover, NH: Amos Tuck School of Business Administration*. Dartmouth College. https://books.google.com/books?id=BNJTAAAAMAAJ

[7] Hendel, Y., & Sourd, F. (2007). An improved earliness-tardiness timing algorithm. *Computers and operations research*, *34*(10), 2931–2938. DOI: 10.1016/j.cor.2005.11.004

[8] Potts, C. N., & Kovalyov, M. Y. (2000). Scheduling with batching: a review. *European journal of operational research*, *120*(2), 228–249. DOI: 10.1016/S0377-2217(99)00153-8

[9] Crauwels, H. A. J., Potts, C. N., Van Oudheusden, D., & Van Wassenhove, L. N. (2005). Branch and bound algorithms for single machine scheduling with batching to minimize the number of late jobs. *Journal of Scheduling*, *8*(2), 161–177. DOI: 10.1007/s10951-005-6365-4

[10] Lee, C. Y., Uzsoy, R., & Martin Vega, L. A. (1992). Efficient algorithms for scheduling semiconductor burn-in operations. *Operations research*, *40*(4), 764–775. DOI: 10.1287/opre.40.4.764

[11] Li, C. L., & Lee, C. Y. (1997). Scheduling with agreeable release times and due dates on a batch processing machine. *European journal of operational research*, *96*(3), 564–569. DOI: 10.1016/0377-2217(95)00332-0

[12] Ikura, Y., & Gimple, M. (1986). Efficient scheduling algorithms for a single batch processing machine. *Operations research letters*, *5*(2), 61–65. DOI: 10.1016/0167-6377(86)90104-5

[13] Cheng, T. C. E., & Kahlbacher, H. G. (1993). Scheduling with delivery and earliness penalties. *Asia-pacific j. opertuin reseach.*, *10*, 145–152.

[14] Cheng, T. C., & Gordon, V. S. (1994). Batch delivery scheduling on a single machine. *Journal of the operational research society*, *45*(10), 1211–1215. DOI: 10.1057/jors.1994.191

[15] Hall, N. G., & Potts, C. N. (2005). The coordination of scheduling and batch deliveries. *Annals of operations research*, *135*(1), 41–64. DOI: 10.1007/s10479-005-6234-8

[16] Pundoor, G., & Chen, Z. L. (2005). Scheduling a production-distribution system to optimize the tradeoff between delivery tardiness and distribution cost. *Naval Research Logistics*, *52*(6), 571–589. DOI: 10.1002/nav.20100

[17] Hall, N. G., & Potts, C. N. (2003). Supply chain scheduling: Batching and delivery. *Operations research*, *51*(4), 566–584. DOI: 10.1287/opre.51.4.566.16106

[18] Selvarajah, E., & Steiner, G. (2006). Batch scheduling in a two-level supply chain-a focus on the supplier. *European journal of operational research*, *173*(1), 226–240. DOI: 10.1016/j.ejor.2004.12.007

[19] Shabtay, D. (2010). Scheduling and due date assignment to minimize earliness, tardiness, holding, due date assignment and batch delivery costs. *International journal of production economics*, *123*(1), 235–242. DOI: 10.1016/j.ijpe.2009.08.012

[20] Benmansour, R., Allaoui, H., Artiba, A., & Hanafi, S. (2014). Minimizing the weighted sum of maximum earliness and maximum tardiness costs on a single machine with periodic preventive maintenance. *Computers and operations research*, *47*, 106–113. DOI: 10.1016/j.cor.2014.02.004

[21] Rostami, M., Kheirandish, O., & Ansari, N. (2015). Minimizing maximum tardiness and delivery costs with batch delivery and job release times. *Applied mathematical modelling*, *39*(16), 4909–4927. DOI: 10.1016/j.apm.2015.03.052

[22] Li, Z., Chen, H., Xu, R., & Li, X. (2015). Earliness-tardiness minimization on scheduling a batch processing machine with non-identical job sizes. *Computers and industrial engineering*, *87*, 590–599. DOI: 10.1016/j.cie.2015.06.008

[23] Ahmadizar, F., & Farhadi, S. (2015). Single-machine batch delivery scheduling with job release dates, due windows and earliness, tardiness, holding and delivery costs. *Computers and operations research*, *53*, 194–205. DOI: 10.1016/j.cor.2014.08.012

[24] Yuan, J. (1996). A note on the complexity of single-machine scheduling with a common due date, earliness-tardiness, and batch delivery costs. *European journal of operational research*, *94*(1), 203–205. DOI: 10.1016/0377-2217(95)00168-9

[25] Kovalyov, M. Y. (1997). Batch scheduling and common due date assignment problem: an NP-hard case. *Discrete applied mathematics*, *80*(2–3), 251–254. DOI: 10.1016/S0166-218X(97)00092-9

[26] Hall, N. G., Sethi, S. P., & Sriskandarajah, C. (1991). On the complexity of generalized due date scheduling problems. *European journal of operational research*, *51*(1), 100–109. DOI: 10.1016/0377-2217(91)90149-P

[27] Garey, M. R., Tarjan, R. E., & Wilfong, G. T. (1988). One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of operations research*, *13*(2), 330–348. DOI: 10.1287/moor.13.2.330

[28] Wan, G., & Yen, B. P. C. (2002). Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European journal of operational research*, *142*(2), 271–281. DOI: 10.1016/S0377-2217(01)00302-2

[29] Chang, P. C., Chen, S. S., & Fan, C. Y. (2008). Mining gene structures to inject artificial chromosomes for genetic algorithm in single machine scheduling problems. *Applied soft computing journal*, *8*(1), 767–777. DOI: 10.1016/j.asoc.2007.06.005

[30] Allaoua, H., & Osmane, I. (2010). Variable parameters lengths genetic algorithm for minimizing earliness-tardiness penalties of single machine scheduling with a common due date. *Electronic notes in discrete mathematics*, *36*(C), 471–478. DOI: 10.1016/j.endm.2010.05.060

[31] Lin, S. W., Chou, S. Y., & Chen, S. C. (2007). Meta-heuristic approaches for minimizing total earliness and tardiness penalties of single-machine scheduling with a common due date. *Journal of heuristics*, *13*(2), 151–165. DOI: 10.1007/s10732-006-9002-2

[32] Kashan, A. H. (2009). League championship algorithm: a new algorithm for numerical function optimization. *In 2009 international conference of Soft Computing and Pattern Recognition* (pp. 43-48). IEEE. DOI: 10.1109/SoCPaR.2009.21

[33] Alimoradi, M. R., & Husseinzadeh Kashan, A. (2018). A league championship algorithm equipped with network structure and backward Q-learning for extracting stock trading rules. *Applied soft computing journal*, *68*, 478–493. DOI: 10.1016/j.asoc.2018.03.051

[34] Bouchekara, H. R. E. H., Abido, M. A., Chaib, A. E., & Mehasni, R. (2014). Optimal power flow using the league championship algorithm: a case study of the Algerian power system. *Energy conversion and management*, *87*, 58–70. DOI: 10.1016/j.enconman.2014.06.088

[35] Abdulhamid, S. M., & Latiff, M. S. A. (2017). A checkpointed league championship algorithm-based cloud scheduling scheme with secure fault tolerance responsiveness. *Applied soft computing journal*, *61*, 670–680. DOI: 10.1016/j.asoc.2017.08.048

[36] Kashan, A. H., & Karimi, B. (2010). A new algorithm for constrained optimization inspired by the sport league championships. *2010 IEEE World Congress on computational intelligence, wcci 2010 - 2010 IEEE Congress on evolutionary computation*, CEC 2010. DOI: 10.1109/CEC.2010.5586364

[37] Husseinzadeh Kashan, A. (2011). An efficient algorithm for constrained global optimization and application to mechanical engineering design: League championship algorithm (LCA). *CAD computer aided design*, *43*(12), 1769–1792. DOI: 10.1016/j.cad.2011.07.003

[38] Pourali, Z., & Aminnayeri, M. (2011). A novel discrete league championship algorithm for minimizing earliness/tardiness penalties with distinct due dates and batch delivery consideration. *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*. Berlin, Heidelberg: springer Berlin heidelberg. DOI: 10.1007/978-3-642-24728-6_19

[39] Ruiz, R., Maroto, C., & Alcaraz, J. (2006). Two new robust genetic algorithms for the flowshop scheduling problem. *Omega*, *34*(5), 461–476. DOI: 10.1016/j.omega.2004.12.006

[40] Abdul-Razaq, T. S., & Potts, C. N. (1988). Dynamic programming state-space relaxation for single-machine scheduling. *Journal of the operational research society*, *39*(2), 141–152. DOI: 10.1057/jors.1988.26

[41] Atashpaz-Gargari, E., & Lucas, C. (2007). Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. *2007 IEEE Congress on Evolutionary Computation*, cec 2007. DOI: 10.1109/CEC.2007.4425083